PERGAMON Mathematical and Computer Modelling 0 (2004) 1–0
www.elsevier.com/locate/mcm

Modelling Force Deployments from Army Installations using the Transportation System Capability (TRANSCAP) Model: A Standardized Approach

J. F. Burke, Jr.*, R. J. Love and C. M. Macal
Argonne National Laboratory
9700 S. Cass Ave, Bldg. 900
Argonne, IL 60439, U.S.A.
<jay><love><macal>@anl.gov

Abstract—Argonne National Laboratory (Argonne) developed the transportation system capability (TRANSCAP) model to simulate the deployment of forces from Army bases, in collaboration with and under the sponsorship of the Military Transportation Management Command Transportation Engineering Agency (MTMCTEA). TRANSCAP's design separates its pre- and post-processing modules (developed in Java) from its simulation module (developed in MODSIM III®). This paper describes TRANSCAP's modelling approach, emphasizing Argonne's highly detailed, object-oriented, multilanguage software design principles. Fundamental to these design principles is TRANSCAP's implementation of an improved method for standardizing the transmission of simulated data to output analysis tools and the implementation of three Army deployment/redeployment community standards, all of which are in the final phases of community acceptance. The first is the extensive hierarchy and object representation for transport simulations (EXHORT), which is a reusable, object-oriented deployment simulation source code framework of classes. The second and third are algorithms for rail deployment operations at a military base.

Keywords—Deployment, Transportation, Discrete-event simulation, Army standards, Standardization, Object-oriented analysis, Object-oriented design.

1. INTRODUCTION

Logistics and mobility have become increasingly important in our rapidly changing world. Since September 11, 2001, it has become even more critical to move soldiers and supplies with limited resources in support of the war on terrorism. During a military deployment, both personnel and

This work was supported under a military interdepartmental purchase request from the U.S. Department of Defense, MTMCTEA, through the U.S. Department of Energy contract W-31-109-ENG-109. The authors wish to acknowledge the support of our program managers of MTMCTEA. The authors also thank members of Argonne's Information and Publishing Division for their editorial assistance.
*Author to whom all correspondence should be addressed.

0895-7177/04/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII:00

equipment are transported from a fort to a tactical assembly area (the logistics and mobility phase) and then to their final destination—the forward line of troops (the war-fighting phase). With so many active deployments in recent years, there has been little opportunity to practice and test experimental ideas, which would not be under realistic circumstances in any case. In these recent active deployments, the Army has had ample time to deploy its forces. However, rapid deployments to suddenly erupting world hot spots remain in the mission statements of many of our most important fighting forces. As a result, simulation continues to be an important tool in preparing for the constraints and complexities of deployments.

To address this issue, Argonne National Laboratory (Argonne), in collaboration with the Military Traffic Management Command Transportation Engineering Agency (MTMCTEA) and others, is developing simulations that may be used to analyze, plan, train for, and execute the transportation portion (part of the logistics and mobility phase) of military deployments. New simulation models and technologies are a good investment because simulating deployment scenarios on a computer is more efficient and cost-effective than testing them in the real world. The transportation system capability (TRANSCAP) model simulates the first step of a military deployment after mobilization, the loading operations at power projection platforms, which are the Army's biggest and best developed home forts in the continental United States. Specifically, the model simulates installation transportation operations, computes the time-phased outloading capability, and identifies system and infrastructure constraints and installation-specific force departure profiles.

This paper describes TRANSCAP's modelling approach, both with regard to the simulation code and the other critical pieces that make for a functioning application. At the heart of the design of the simulation code are classes with fields and methods specific to Army installations, as part of its implementation of a reusable deployment simulation framework of classes [1]. This framework, called the extensive hierarchy and object representation for transport simulations (EXHORT), was created as part of the design for TRANSCAP and has been successfully reused in the coastal integrated throughput model (CITM), another MTMCTEA and Argonne deployment simulation, thus proving the general utility of the design, since CITM and TRANSCAP simulate radically different processes. The other modules of TRANSCAP were designed to use other Army deployment community standards, and were organized into distinct modules operating as a cooperative cluster. This layer of abstraction allowed us to reap the benefits of modern software design strategies.

2. MODELLING APPROACH

TRANSCAP was written at the lowest possible level of data and process. Its processes were also written in an object-oriented programming language to foster the greatest amount of modularity and potential for reuse for future simulation projects.

The following paragraphs describe the three types of low-level data that TRANSCAP uses: transportation assets and cargo, infrastructure, and resource. Transportation and cargo data are at the maximum level of detail, Level 6, which means that details are available for each piece of cargo and each transport vehicle. These are each configured for their particular mode of movement, rather than being an aggregated data description based on tonnage or category. Infrastructure data is the most detailed type of data concerning the quantity and measurements of existing infrastructure. For example, rather than using an aggregate capacity for all marshaling areas or rail yards, the model uses specific square footage and tangent spur-length data for each. Resource data include information about existing facility resources such as the quantity and location of truck loading ramps, rail end ramps, inspectors, mechanics, and tie-down crews [2].

This highly detailed data drove the design of the application, as well as the simulation. In deployment simulations, there are also two types of processes—infrastructure utilization (movement) and resource utilization (a specific activity). In TRANSCAP, because of the detail of the

data, these processes are modelled at a high level of detail. Processes simulating infrastructure utilization are based on the actual distances that are traversed in the infrastructure, rather than on aggregate stochastic distributions. Resource processes performed on transportation assets and cargo are associated with individual resources, and each individual resource is represented as an object. Again, there are no unknown actors performing aggregate processing on unknown entities. Each resource is assigned a specific responsibility and a specific service time, instead of representing multiple processes with a general rate. The design of the simulation is explained in detail in Section 4.

The highly detailed data also drove the design of the application. High resolution typically creates a large quantity of data to be handled between modules of the application, and between applications in a federation. Details of these design issues are explained in Sections 5 and 6.

3. IMPLEMENTING THE COMPONENT MODULES IN DIFFERENT LANGUAGES

TRANSCAP comprises four categories of component modules: preprocessing, simulation, post-processing, and interprocess communication.

- The modules of the preprocessing category include the installation, force, and scenario manager and the scenario report manager. These preprocessing modules manage data input, create the installation, select the installation, specify pieces for deployment, identify the method of deployment (e.g., convoy), and specify deployment characteristics (e.g., the method used to form the convoy groups). Most of these tasks are encapsulated in the user interface, which also can be thought of as a component module.
- The simulation category contains the heart of TRANSCAP, the discrete-event simulation module.
- The modules of the postprocessing category handle the reporting, presentation, animation, and analysis of simulation results and the export of results to other simulation models.
- The module of the interprocess communication (IPC) category enables interaction between the pre- and post-processing modules and the simulation module.

Due to its portability across the multiple platforms employed by Army analysts, Java was the natural choice for the main programming language for all of these component modules. However,

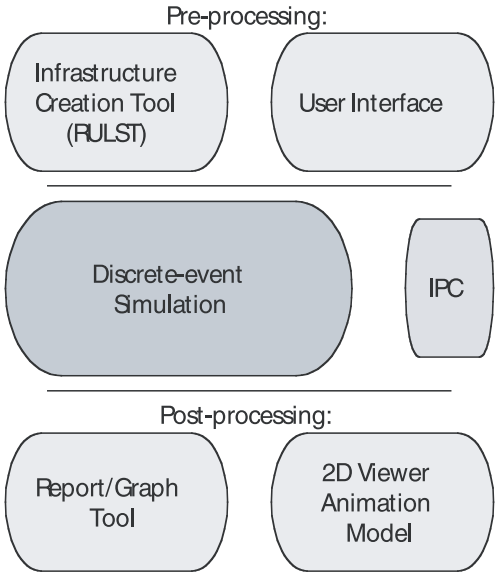

Figure 1.

we decided to implement the actual discrete-event simulation in (recently sold by CACI to) Compuware's MODSIM III® programming language, a language that is designed for the special needs of discrete-event programming, and which is a much more suitable tool for this type of programming than Java has been during the years of the development of this model.

4. SOFTWARE DESIGN OF THE DISCRETE EVENT SIMULATION

This section describes the layered approach taken in designing the discrete-event simulation. EXHORT, a reusable deployment simulation class framework, is subclassed under the strategy design pattern, thus creating another class framework encapsulating Army installation specifics. This additional class framework provides the building blocks from which military processes are encapsulated.

4.1. The EXHORT Framework

EXHORT consists of two hierarchies that together constitute a standard and consistent class attribute representation and behavior that could be used directly by a large set of deployment simulations. The hierarchies are unusual in that they provide for a class factoring detailed enough to be a document from the design workflow, rather than merely an aggregate document from the analysis workflow. To encapsulate the characteristics of multiple deployment simulations written at different levels of aggregation, the class abstractions in EXHORT needed to be presented as a design workflow document, rather than as an analysis workflow document, which is more typical of object modeling in the larger community. This greater detail was incorporated because we decided that these classes were "key abstractions" and are well suited for use in new federations of simulations where reusability and intercommunication for new simulations are important [3].

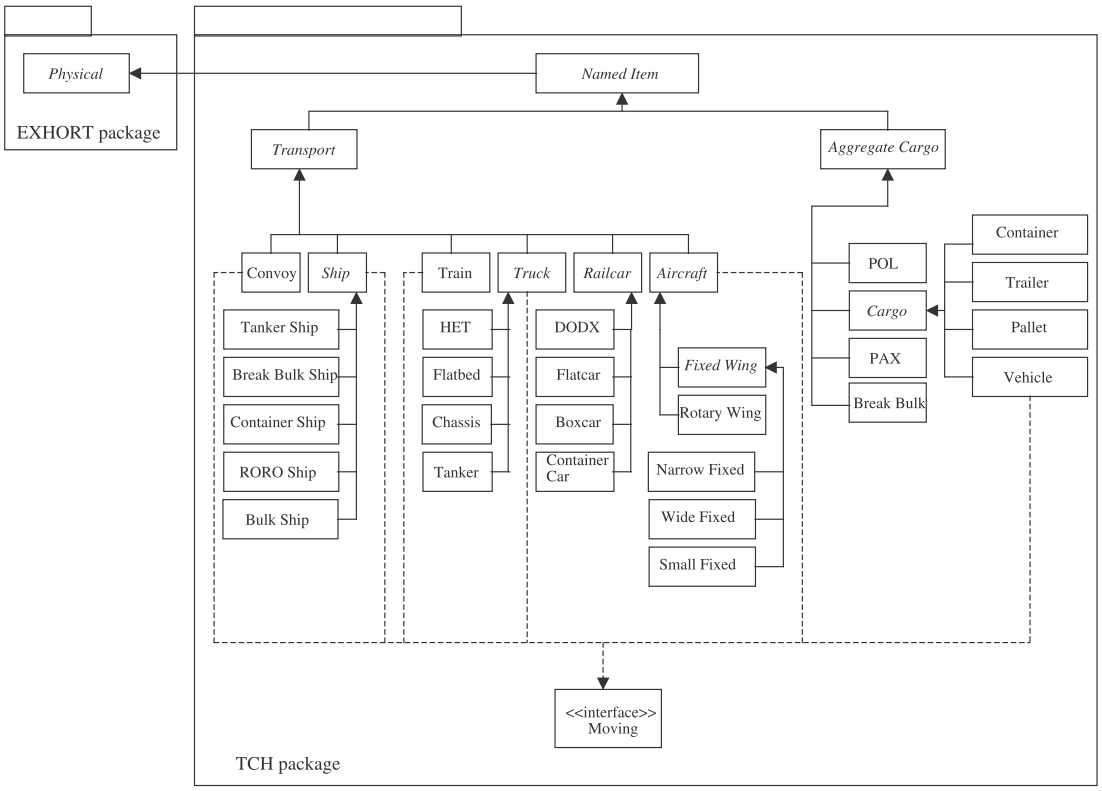

Figure 2. Transportation class hierarchy.

The two EXHORT hierarchies will allow many deployment simulations to use the same set of underlying class data and will significantly reduce the effort needed to integrate simulations that analyze the defense transportation system. The first hierarchy, the transportation class hierarchy (TCH), encapsulates commercial transportation assets and military cargo. This hierarchy is the Army Modeling and Simulation Office (AMSO) deployment/redeployment community standard SRD 00068 in AMSO's Army standard repository (ASTARS), at `http://www.msrr.army.mil/astars` [3].

The second hierarchy is the infrastructure class hierarchy (ICH), which deals with locations where activities are performed and with the physical infrastructure, such as a motor pool at a fort or a berth at a port. These areas contain the resources needed to support deployment (e.g., ramps, rough-terrain container handlers, cranes, and forklifts). Both hierarchies are needed to describe major portions of the defense transportation system. The goal was to define EXHORT so it could provide a standardized code structure for object-oriented deployment simulations to ensure meaningful data exchanges. A report on the ICH was also submitted to AMSO's deployment/redeployment community for acceptance in the ASTARS repository [3].

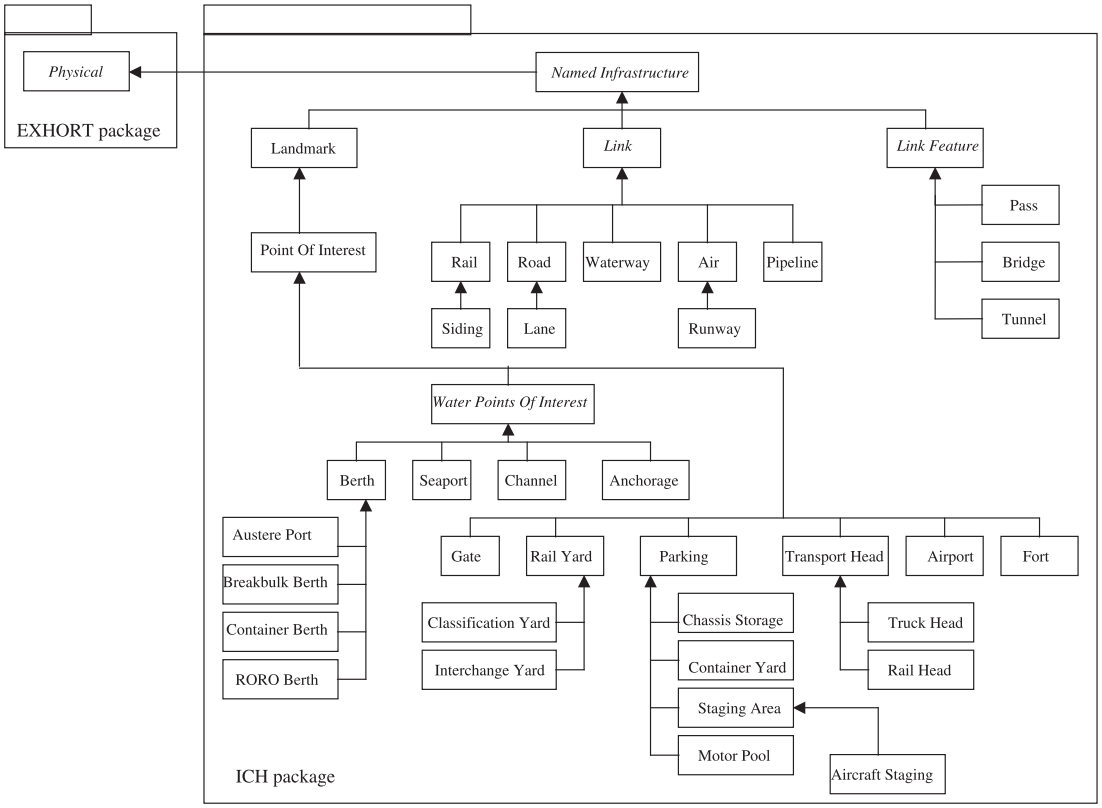

Figure 3. Infrastructure class hierarchy.

4.2. Army Installation-Specific Subclasses of the EXHORT Framework

EXHORT was extended for TRANSCAP to develop military fort-specific subclasses containing logic for specific interactions between clients, servers, and areas. In sum, these subclasses form a new class framework composed of specific "business logic" of military fort deployment that overrides certain generic deployment simulation logic, as needed. An example of how the strategy design pattern is implemented to create this new framework of classes is a vehicle being loaded onto a railcar. In the reusable framework, the interaction between clients and servers is implemented as a resource holding a piece of cargo in place for a stochastic process time and then updating the vehicle's current status field when completed. No additional processing is done to

the client or the server during the service period. However, when a vehicle (i.e., a client) interacts with a rail end ramp (i.e., a resource) with the goal of loading that vehicle onto the next available position on a train of railcars, the code that chooses a process time stochastically needs to be overridden. The new code (that overrides the old code) will choose a process time on the basis of the length of the railcars and spanners over which the vehicle must drive. In this way, the generic interaction between a client and a server, which is found in the reusable deployment simulation, is subclassed for a specific situation at a fort [2].

4.3. Building Military Processes

The EXHORT framework, which is the kernel of the TRANSCAP discrete-event simulation, was subclassed to create interactions specific to an Army installation deployment simulation. This new class framework inherits and, when appropriate, overrides EXHORT attributes and interactions. This new Army installation-specific framework consists of components that are the building blocks of TRANSCAP's discrete-event simulation processes. The ten processes created from these building blocks include:

- simulation initialization,
- vehicle generation,
- vehicle processing,
- transport loading,
- container movement,
- truck transport asset generation,
- truck transport asset processing,

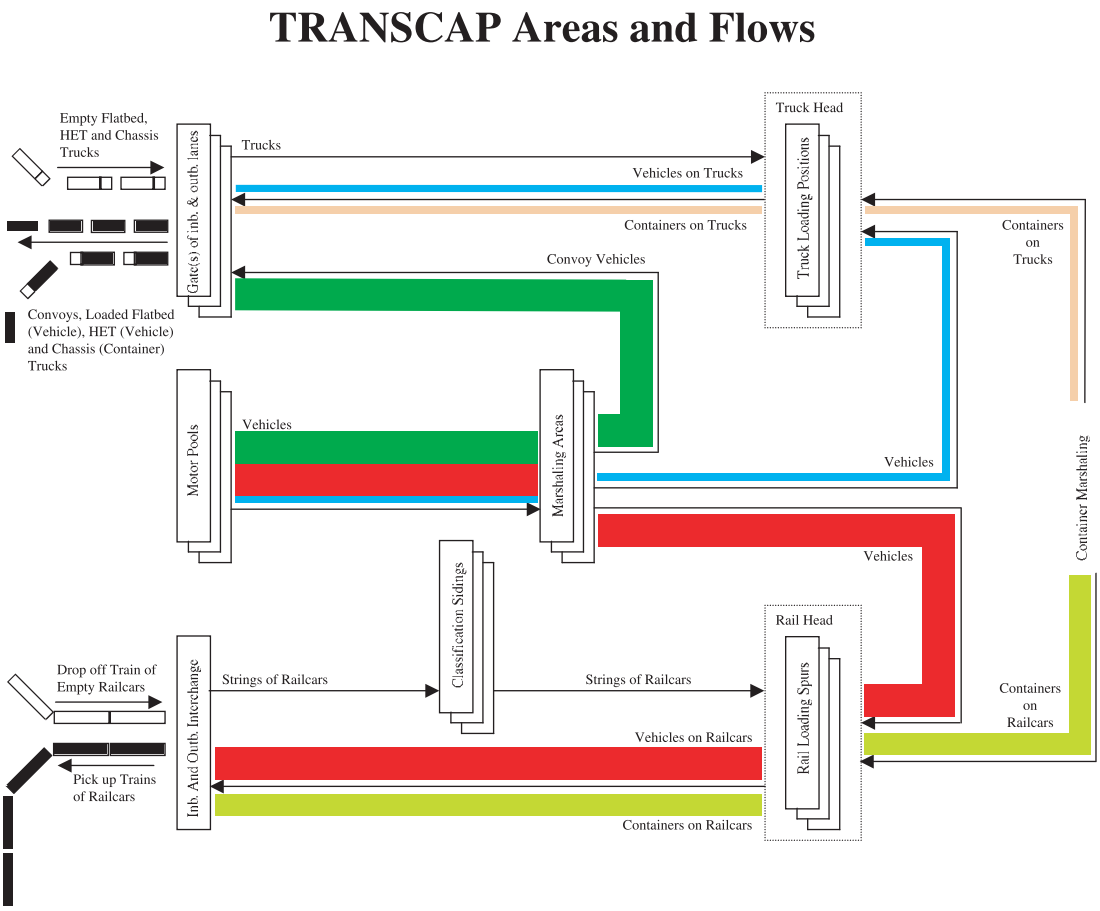

Figure 4.

- train generation,
- train processing,
- call forward locomotive processing.

As with EXHORT, these ten processes that comprise an Army installation class framework can be reused by any deployment simulation that requires any of these Army installation processes. The transport loading and train processing processes were designed in the greatest detail. As such, they were submitted to AMSO's deployment/redeployment community for acceptance in the ASTARS repository [4].

When simulating, these military processes execute in parallel to produce a scenario of cargo flows through an Army fort, as pictured in Figure 4. Military vehicles are assigned (by the user or an input source) to leave the fort by convoy, by highway on a truck, or by rail. These vehicles start at their home motor pool and begin by moving to a marshaling area for various inspections and repairs, and then onto a gate (if leaving by convoy) or to the appropriate loading area. Containers start at a container marshaling area, and are called forward to an appropriate loading area. (Container preparation is currently not modeled.) Truck transport assets start with gate processing, followed by a movement to a truck loading position, where they wait to be loaded. Loaded trucks then return to their original gate with cargo. A train of railcars, dropped off by a commercial carrier, start with interchange siding processing, followed by a movement to classification sidings, where they are sorted. At these sidings, strings of railcars are called forward to loading spurs, based on cargo ready for loading at that simulation time. Loaded railcars are then taken back to an interchange siding before being removed from the installation by a commercial carrier.

5. ARCHITECTURE OF COMPONENT MODULES

A kind of cluster architecture is used for TRANSCAP's component modules. Instead of one monolithic simulation model that also handles all preprocessing, postprocessing, and communication between these processes, TRANSCAP's capabilities have been separated into distinct modules and arranged as a cooperative cluster (see Figure 1) [4]. This provides a layer of abstraction that allows each module to

(1) focus on its own tasks,
(2) be written in a language that is appropriate for its tasks,
(3) be developed independently, and
(4) more easily take advantage of a multiple processor or a distributed computing environment.

Further, other models can then more easily reuse TRANSCAP's modules, and vice versa. In fact, reuse is already being achieved with several modules; other simulations also use the RULST (route and landmark selection tool) infrastructure creation tool, the report/graph tool, and the 2D viewer animation model.

With this type of architecture, however, the data exchanged between modules tends to increase in volume, so standardizing the data mapping becomes more important. This data exchange becomes the interface between the modules, as well as between TRANSCAP and other models. The format of this data becomes vital when multiple models and modules read some of the same data files.

6. STANDARDIZATION BETWEEN MODULES

As the interface between modules and models, data exchange can benefit greatly from standardization. By standardizing the format of a data file, the data becomes easier to understand. By leveraging standard tools (e.g., APIs, XML), the development of software code is alleviated. In both cases, it becomes much easier to share the data and reuse file formats, processes, com-

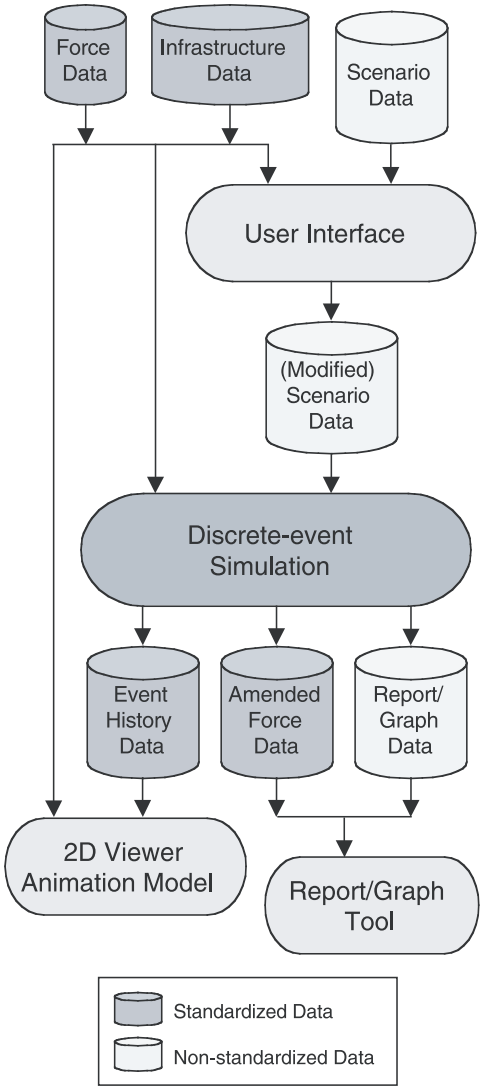

Figure 5. Data file flow.

ponents, and code. To various degrees, TRANSCAP takes advantage of both standardized data file formats and tools, and more standardization efforts are planned for the future.

The data files used by TRANSCAP, standardized or not, are shown in Figure 5, as well as how they flow through the model.

6.1. Force Data

Standardization of force data is critical among deployment simulations that will be participating in the same federation. Both infrastructure and scenario data are within the boundaries of what TRANSCAP is responsible for simulating, and do not necessarily need to be reusable across applications. However, the actual cargo items that make up the deploying forces must be passed between the various applications that simulate the different legs of a deployment. Therefore, multiple deployment applications will be using the same data, and should use the same data sources. As such, TRANSCAP reuses a standard force profile database to determine which cargo objects will be simulated. A standard file format for a subset of that information is then reused among TRANSCAP's modules and between TRANSCAP and other models.

MTMCTEA and Argonne also collaborated to create the expanded TPFDD editor (ETEdit) and the theater-level analysis, replanning, and graphical extension toolbox (TARGET) force

database systems to convert the nearly universal, aggregated time phased force deployment data (TPFDD) into highly detailed Level 6 data. TRANSCAP and the other highly detailed deployment simulations then use this converted data, via standard, unclassified type unit characteristics (TUCHA) force templates. The end data produced for TRANSCAP describes each piece of cargo (e.g., wheeled or tracked vehicles, trailers, and stuffed containers) of each unit (TRANSCAP requires that data be specified at the company and small battalion level) located at a particular military fort, as well as a TARGET-optimized deployment mode (e.g., leaving the fort by road in a convoy, loaded on a truck transport, or loaded on a train). Different data sets are produced for different deployment mode optimizations, with the same cargo in the same units.

6.2. Infrastructure Data

The infrastructure data was designed to be independent of the simulation models that use them. They encapsulate data about the physical structures of Army forts, seaports, and airports, such as landmarks and the road, rail, sea, and air routes between them, including their geographic location. RULST is a graphical map-based model that facilitates the identification and creation of this data. It also provides a way to add logical data on top of the physical data, such as the availability and direction of routes. Any number of specific instances of these installations can then be saved as a standardized set of infrastructure files.

Since the infrastructure database itself was designed with the needs of various deployment simulations in mind, it was also submitted and accepted (SRD 00152) by the AMSO deployment/redeployment community as a standard database for the community. A common infrastructure database allows deployment simulations to use an identical set of underlying infrastructure data. This significantly reduces the effort required for both maintaining the common infrastructure data and integrating the defense transportation simulations for analysis. A relational database is chosen as the proposed standard because it is virtually a *de facto* standard in the larger community, and therefore, is able to support the greatest possible number of deployment simulations. By using more recent database design concepts, such as object-oriented databases, it would be a more difficult fit for older and perhaps non-object-oriented simulations. The goal is to provide a standardized infrastructure database for deployment simulations as a way to ensure meaningful data exchanges between all models in the deployment community [5].

For the infrastructure database, an area of interest is called a landmark. A landmark might be a structure (e.g., gate) or an area of land (e.g., staging area); that is, a landmark is any place where activities occur. Attributes are defined for each landmark that uniquely identify it throughout the entire database, which may contain multiple facilities. Identification attributes are common to all infrastructure tables. For the gate, only one attribute needs to be obtained to determine how many vehicles can be processed simultaneously—the number of inbound lanes. For the staging area, only one attribute needs to be obtained to determine if there is sufficient space to park arriving vehicles—the total available area. Some of the attributes of landmarks required by deployment simulations are not built into the infrastructure database, since they do not pertain to physical and spatial infrastructure [5].

Deployment simulations consistently use certain types of infrastructure data, such as the lengths and positions of a road and rail network, or the distances between two significant facilities (i.e., forts and ports) or landmarks (e.g., parking lots, call forward lines in front of a loading ramp, spurs, or gates—locations used during a deployment). The data requirements supporting deployment simulations extend beyond the physical and spatial infrastructure of the facility. Types of equipment, such as truck end ramps or wharf cranes, are used for facility operations. The physical and spatial attributes of such equipment are available and incorporated into the database; the resource-oriented attributes (e.g., the number of minutes per inspector per inspection) are not [5].

The standardization of these sets of infrastructure files allows RULST to provide data for several of our models. TRANSCAP uses sets of files that describe Army forts. Our port simulation

(PORTSIM) model uses those that describe seaports. Our enhanced logistics intratheater support tool (ELIST) uses those that describe the networks of routes between forts, seaports, and airports. TRANSCAP's 2D viewer animation module also uses the same set of files to graphically play back the event output generated by the simulation module [6]. Further, partially because of this standardization, the report/graph tool module may also be redesigned to take advantage of these sets.

6.3. Scenario Data

A scenario file encapsulates all of the data that TRANSCAP needs to simulate a specific scenario. It includes a pointer to a force file; a small amount of infrastructure data that has been extracted from a set of infrastructure files; and fort-specific deployment data, which includes resource-oriented attributes. The user interface module reads in the scenario data, populating the many windows, and allowing the data to be viewed, edited, and saved in new scenario files. The user interface also directs the simulation module to run a scenario by telling it which scenario file to read.

Infrastructure data extraction is performed due to the small amount of data that had been used within the simulation in the past. As TRANSCAP has developed, however, the simulation has used more of the infrastructure data. Further, planned enhancements regarding how the simulation handles traffic will require even more of the data. Thus, a pointer to a set of infrastructure files will likely replace the extracted data.

6.4. Event History and Report/Graph Data

The beginning, end, or instantaneous completion of an interaction between an object with a lineage from the TCH (e.g., cargo and transportation assets) and a resource object (e.g., cranes, truck and rail end ramps, locomotives, inspectors) at an ICH object (e.g., a staging area, a rail spur) during a TRANSCAP run is called an event. TRANSCAP writes all of these events to an event history database. An event can be conceived of as a database join of information about a client (from the TCH) with a server at an area (from the ICH) at a simulated time. As such, an event record consists of a simulated event, a simulated time, and several keys to the force, infrastructure, and scenario databases.

The simulation module typically generates tens of thousands of events while simulating a given scenario. These results must be output in a way that allows for analysis and presentation, as well as for other modules/models to input the data. Early in TRANSCAP's development, various reports and graphs were the only desired output. So the results as a whole were not saved, but data was aggregated into several files, each one formatted specifically to support a different report or graph. As development continued through its planned phases, the simulation grew more detailed, analysis requirements increased, and more analytical tools were added (specifically, more reports and graphs and new 2D and 3D animation models). This presented a problem; the simulation module should be focused on simulating the deployment scenarios and outputting the results, not on supporting multiple postprocessors.

To solve this problem, the event history was designed as a standard output format for all of the simulation's results, effectively abstracting the postprocessor support. It shields the simulation from dealing with postprocessor changes or additions, while still providing those postprocessors with the required data.

The event history consists of the entire event data created by a simulated scenario, with a header section added to the beginning. The header contains scenario-level data and meta-data about the event data. The scenario-level data includes pointers to the force and infrastructure files that were used to simulate the scenario, further leveraging those standard files. The event history is written as a fixed-column, comma-delimited text file, which is a format that can be

read by almost any program. Java programs can read fixed-column formats quickly, and many spreadsheet and database programs can read comma-delimited formatted data.

The 2D viewer animation module extracts the output data that it needs from this one file, as well as the pointers to where it can find the associated force and infrastructure files. The report/graph tool is being reconfigured to get its data from this one file, instead of many report- or graph-specific files. Spreadsheets and databases can also be used to perform systematic analysis of the simulation.

Because of the large volume of data in an event history, investigations have been made into putting the event data into a database, serializing the events as Java objects, and using XML as the file format. A database is obviously a standard tool; its event data could be analyzed directly, and would still be available to Java programs. Serializing Java event objects would require the definition of a standard event class, which could possibly be used by other Java programs, and would improve event history I/O dramatically. However, it would make analysis more difficult for non-Java programs and tools, such as spreadsheets and databases.

Using XML, a standard data format, would allow TRANSCAP to take advantage of a great variety of existing XML tools, with more on the way. Java programs could use existing XML-handling APIs, and many spreadsheets and databases can now input XML files. Further, as a self-describing format, making changes in the structure or contents of the event history data would be much less likely to require changes in the programs that output or input the data. This should reduce the development time of these programs. For these reasons, the event history is headed for XML.

7. CONCLUSION

TRANSCAP is a significant part of the military deployment community's plan to create an HLA federation of deployment simulations. As such, most interaction with other simulations is expected to be achieved through HLA. Already, the community has begun work on linking the AMP, MIDAS, and ELIST federations. TRANSCAP has a strong connection to ELIST in design, implementation, and data sources. With ELIST as the CONUS (continental United States) model in the AMP federation, TRANSCAP could be used by ELIST as a more focused fort simulation in a running federation [6]. The end goal is a complex, predictive planning tool simulating the entire deployment process. TRANSCAP can also be used independently by civilian installation transportation officers and military division transportation officers for planning a particular installation's deployment. TRANSCAP's architecture was designed to take advantage of the benefits of standardized, reusable deployment simulation classes and installation components and the independent nature of its major processes. Its pre- and post-processing component modules have been separated from the discrete-event simulation to simplify both design and development, which are facilitated by the use of standardized classes and file formats. TRANSCAP's potential for reuse is enhanced by having and sharing well-defined modules and implementing three AMSO deployment community standards: EXHORT, the infrastructure data standard, and the rail loading and switching algorithm candidate standard. This approach has resulted in a richer, more dynamic simulation model.

REFERENCES

1. J.F. Burke, C.M. Macal, M.R. Nevins, C.N. VanGroningen, D.L. Howard and J. Jackson, Standardization of transportation classes for object-oriented deployment simulations, In *Proceedings of the Simulation Interoperability Standards Organization 1999 Fall Simulation Interoperability Workshop*, Orlando, FL, pp. 1142–1152, (1999).
2. J.F. Burke, R.J. Love, C.M. Macal, D.L. Howard and J. Jackson, Modeling force deployment from army installations using the transportation system capability (TRANSCAP) model, In *Proceedings of the 2001 Summer Computer Simulation Conference*, Orlando, FL, pp. 663–668, (2001).
3. J.F. Burke, C.N. VanGroningen, M.J. Bragen and C.M. Macal, Adding the infrastructure class hierarchy to the EXHORT framework for object-oriented deployment simulations, In *Proceedings of the Simulation Interoperability Standards Organization 2002 Fall Simulation Interoperability Workshop*, Orlando, FL, (2002).

4. R.J. Love, J.F. Burke, C.M. Macal, D.L. Howard and J. Jackson, The transportation system capability model (TRANSCAP): A mixed language development approach for an army deployment simulation, In *Proceedings of the 2001 Summer Computer Simulation Conference*, Orlando, FL, pp. 669–674, (2001).
5. M.J. Bragen, J.F. Burke, W.H. Horsthemke, D.L. Howard, J. Jackson, R.J. Love, C.M. Macal, C.N. VanGroningen and M.A. Widing, Standardizing infrastructure data for deployment simulations, Report prepared for MTMCTEA and presented to the Army Modeling and Simulation Office as part of the Army Model Improvement Plan, (1999); Available at `http://www.msrr.army.mil/astars`.
6. R.J. Love, W.H. Horsthemke, J.F. Burke and C.M. Macal, Spatio-temporal animation of an army logistics simulation, the transportation system capability (TRANSCAP) model, In *Proceedings of the 2001 Summer Computer Simulation Conference*, Orlando, FL, pp. 332–337, (2001).